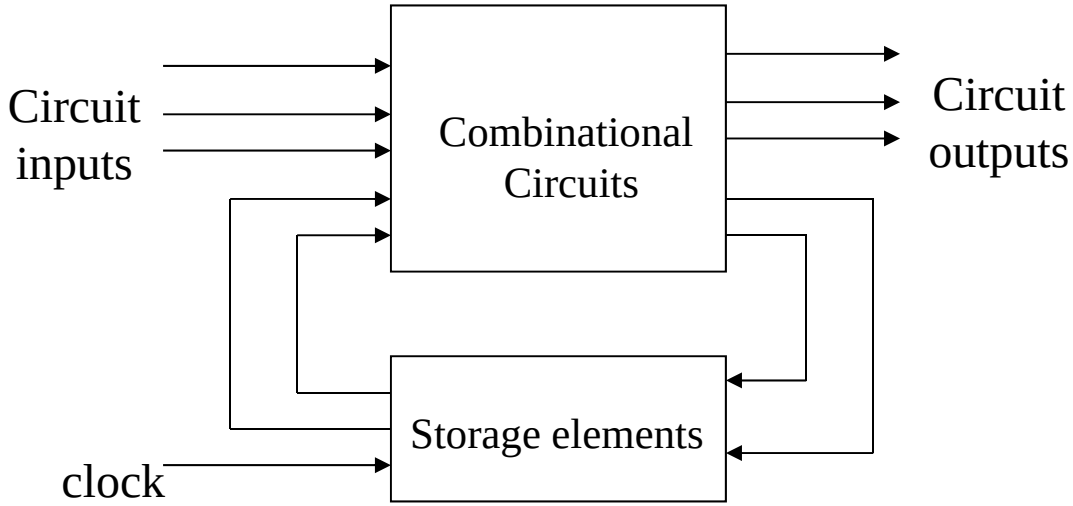
FPGA Implementation of Sequential Logic

Haibo Wang
ECE Department
Southern Illinois University
Carbondale, IL 62901

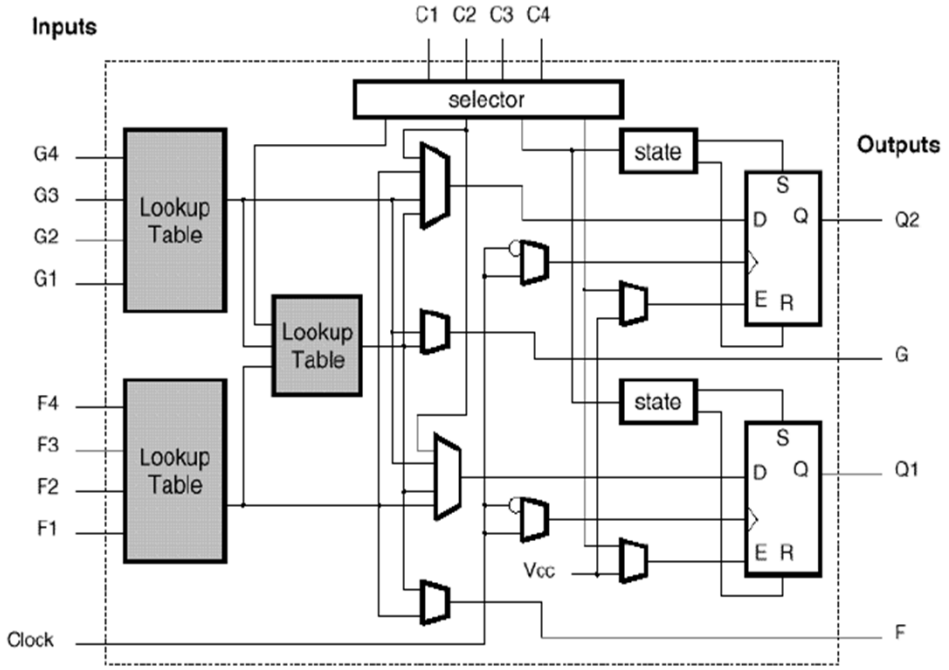
Sequential Circuit Model

- **Combinational Circuit:** the circuit outputs are a logic combination of the current inputs signals.
- **Sequential Circuit:** the circuit outputs depend on not only the current values of inputs but also previous input values.



A model for sequential circuits

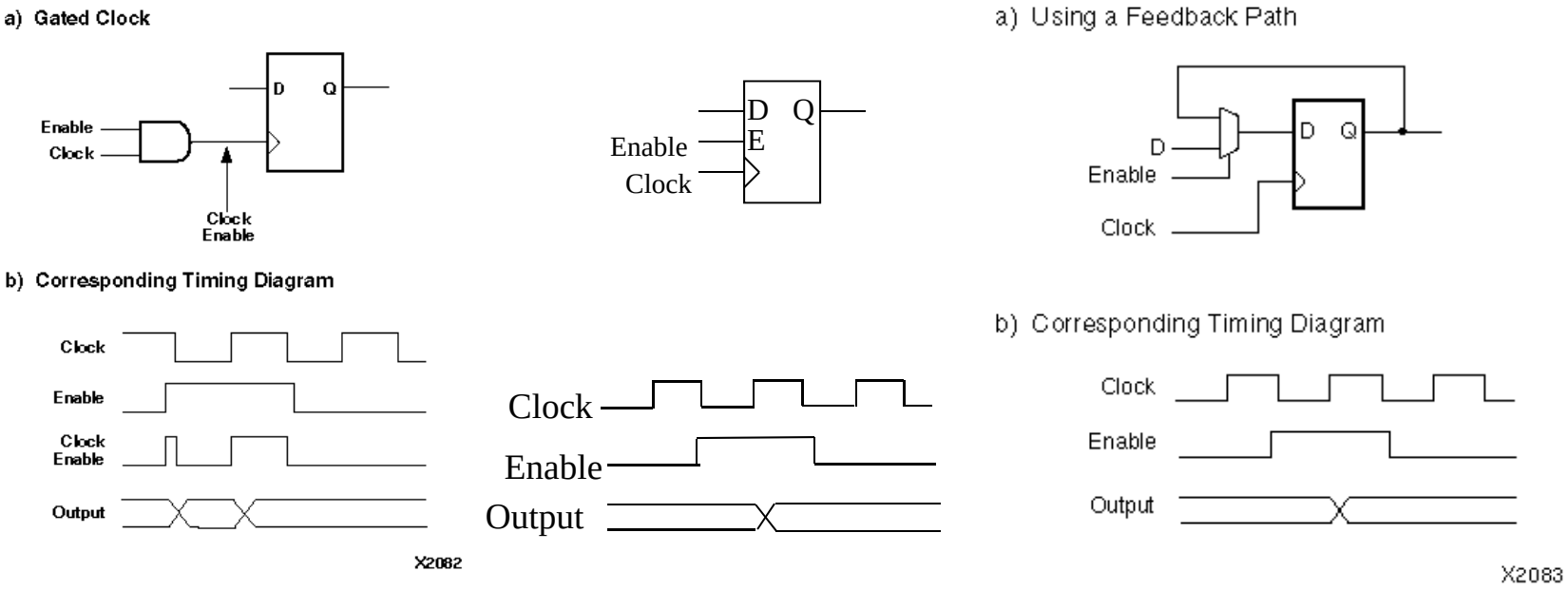
Storage Elements in Xilinx CLB



- Each CLB contains two edge-triggered D flip-flops. They can be configured as positive-edge-triggered or negative-edge-triggered.
- Each D flip-flop has clock enable signal E, which is active high.
- Each D flip-flop can be set or reset by SR signal. A global reset or reset signal is also available for set or reset all D flip-flops once the device is powered up.

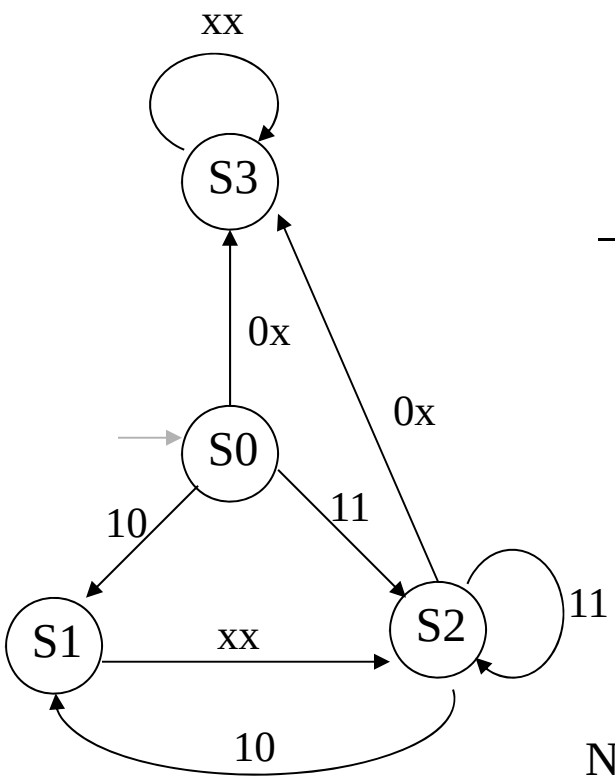
Circuit Techniques to Avoid Clock Glitches

- If possible, try to avoid connecting the output of combinational logic to D flip-flop clock input.



FPGA Implementation of Finite State Machines

- Example of Finite State Machine



State transition diagram

Current States	Inputs: **xy** 0x	10	11	Outputs **a**	**b**	**c**	**d**	**e**
S0	S3	S1	S2	0	0	1	1	1
S1	S2	S2	S2	0	1	0	1	1
S2	S3	S1	S2	1	0	0	1	0
S3	S3	S3	S3	0	0	0	0	0

State Table

Note: this is a Moore-type machine. The design procedure for mealy-type machine is similar.

State Encoding

- **Binary encoding**: minimum number of D flip-flops

	Q1	Q0
S0 :	0	0
S1 :	0	1
S2 :	1	0
S3 :	1	1

It needs two D flip-flps

- **One-hot encoding**: one D flip-flop for each state

	Q3	Q2	Q1	Q0
S0 :	0	0	0	1
S1 :	0	0	1	0
S2 :	0	1	0	0
S3 :	1	0	0	0

It needs four D flip-flps

Implementation Using Binary Encoding

- Excitation table

Inputs		Current States		Next States		Outputs				
x	**y**	Q1	Q0	D1	D0	**a**	**b**	**c**	**d**	**e**
0	x	0	0	1	1	0	0	1	1	1
0	x	0	1	1	0	0	1	0	1	1
0	x	1	0	1	1	1	0	0	1	0
0	x	1	1	1	1	0	0	0	0	0
1	0	0	0	0	1	0	0	1	1	1
1	0	0	1	1	0	0	1	0	1	1
1	0	1	0	0	1	1	0	0	1	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	1	0	0	0	1	1	1
1	1	0	1	1	0	0	1	0	1	1
1	1	1	0	1	0	1	0	0	1	0
1	1	1	1	1	1	0	0	0	0	0

Implementation Using Binary Encoding

- Combinational functions needed to be implemented

$$D1 = x' + y + Q0 \quad \text{(F1)}$$

$$D0 = Q1 \cdot Q0 + y' \cdot Q0' + x' \cdot Q0' \quad \text{(F2)}$$

$$a = Q1 \cdot Q0' \quad \text{(F3)}$$

$$b = Q1' \cdot Q0 \quad \text{(F4)}$$

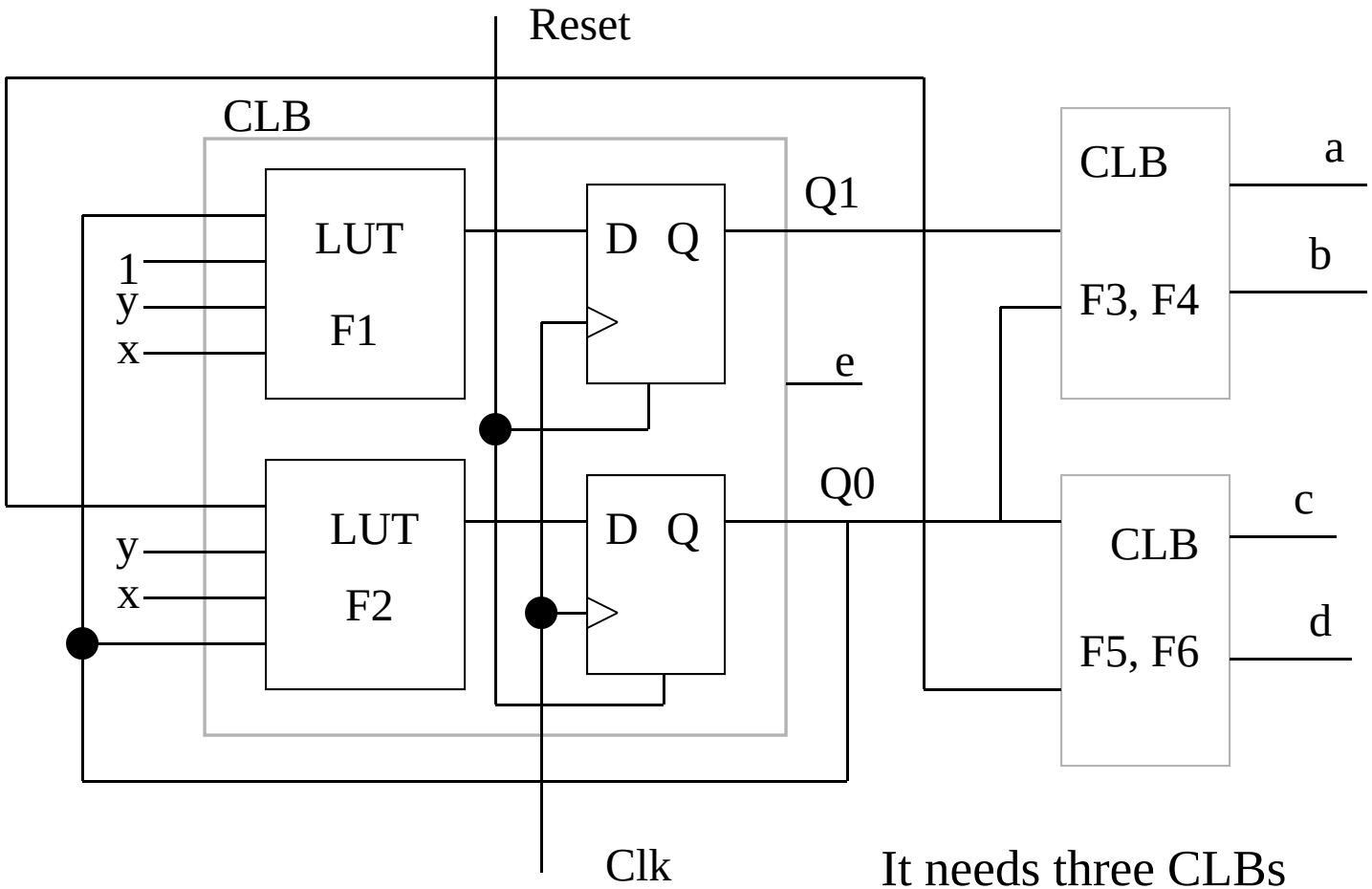
$$c = Q1' \cdot Q0' \quad \text{(F5)}$$

$$d = Q0' + Q1' \quad \text{(F6)}$$

$$e = Q1' \quad \text{(F7)}$$

Implementation Using Binary Encoding

- FPGA implementation



It needs three CLBs

Implementation Using One-Hot Encoding

- The next state and output functions have a simple, systematic form
  - Next state function

$$D_i = \sum Q_j (I_{j,1} + I_{j,2} + \bullet\bullet\bullet + I_{j,n})$$

    - D_i is the input of the D flip-flop that represents state S_i
    - Q_j is the output of the D flip-flop that represent state S_j
    - $I_{j,1}$, $I_{j,2}$, ….. and $I_{j,n}$ denote all input combinations that cause a state transition from S_j to S_i

  - Output function

$$z_k = Q_{k,1} + Q_{k,2} + \bullet\bullet\bullet + Q_{k,m}$$

    - z_i is an FSM output
    - $Q_{k,1}$, $Q_{k,2}$, ….. and $Q_{k,m}$ denote all states (D flip-flip outputs) that cause output z_k to be 1

Implementation Using One-Hot Encoding

- Combinational functions needed to be implemented

$$D0 = 0 \quad \text{(F1)}$$

$$D1 = Q0 \bullet x \bullet y' + Q2 \bullet x \bullet y' \quad \text{(F2)}$$

$$D2 = Q0 \bullet x \bullet y + Q1 + Q2 \bullet x \bullet y \quad \text{(F3)}$$

$$D3 = Q0 \bullet x' + Q2 \bullet x' + Q3 \quad \text{(F4)}$$

$$a = Q2$$

$$b = Q1$$

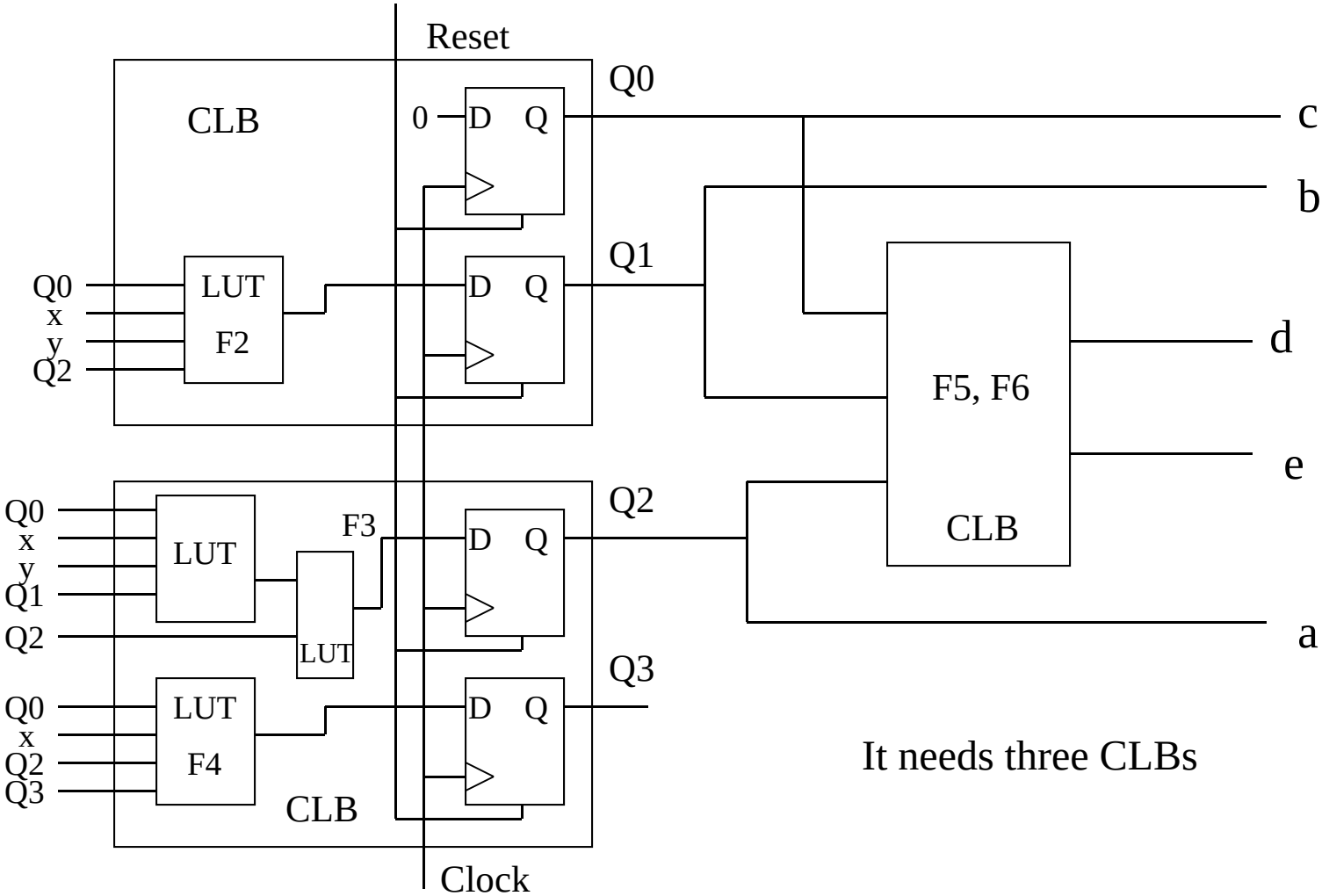
$$c = Q0$$

$$d = Q0+Q1+Q2 \quad \text{(F5)}$$

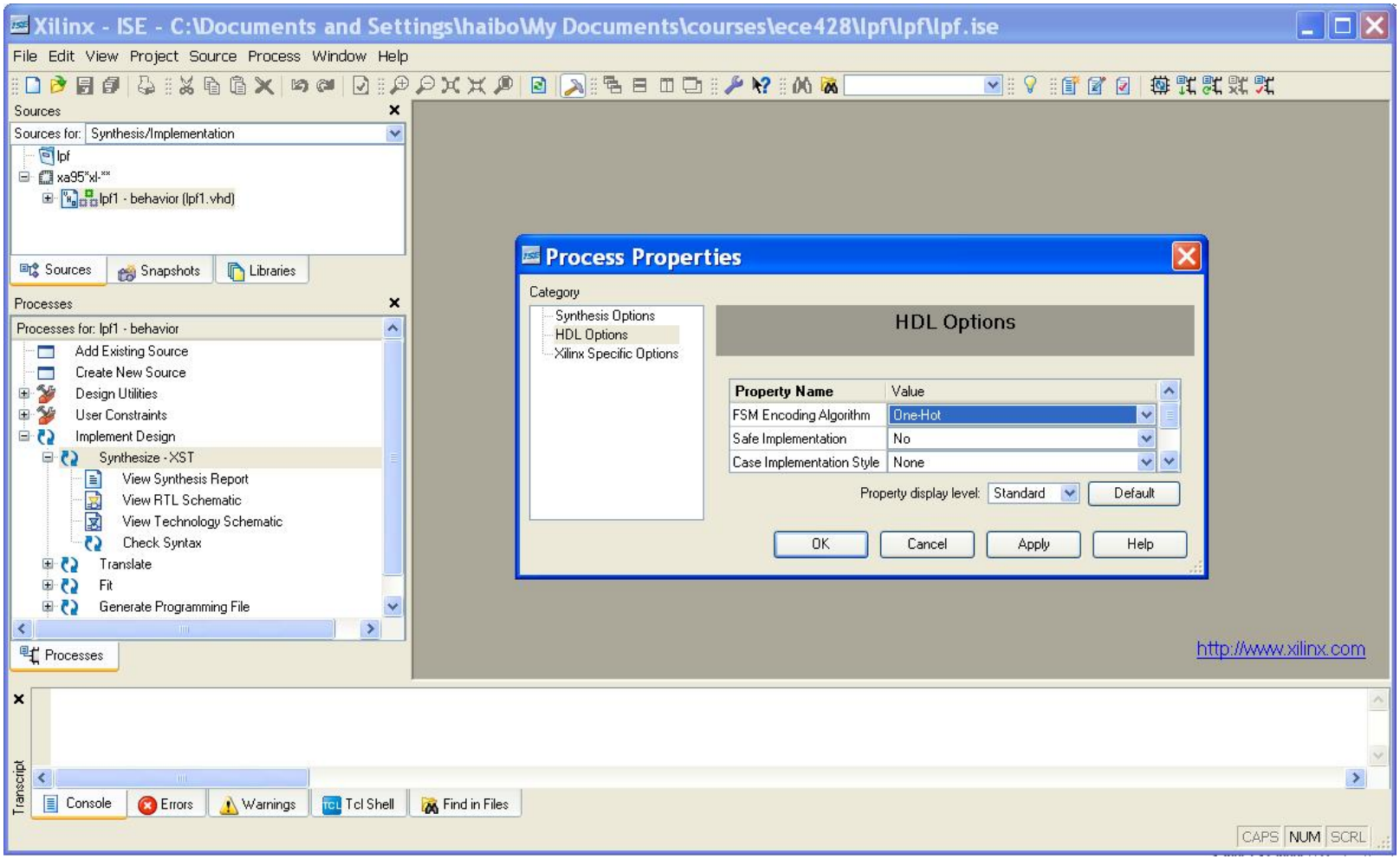
$$e = Q0+Q1 \quad \text{(F6)}$$

Implementation Using One-Hot Encoding

- FPGA implementation



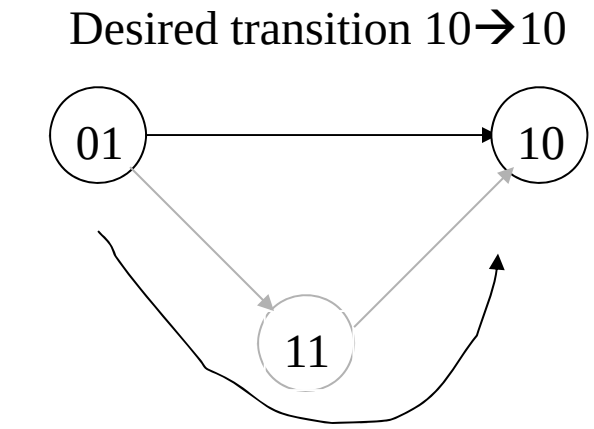
Selecting FSM Coding in Xilinx ISE



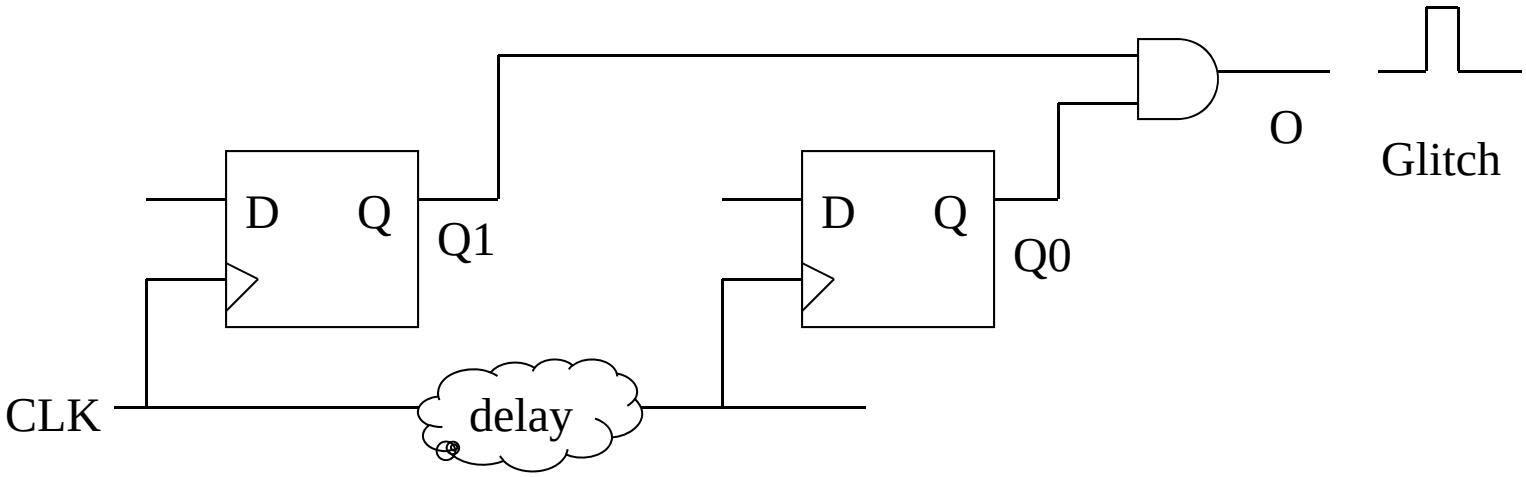
Comparison of Binary Encoding and One-Hot Encoding

- Binary encoding
  - Fewer flip-flops
  - It normally needs complicated combinational logic to determine next state and output signals. The complicated logic may decrease circuit performance.
  - May have glitches
- One-hot encoding
  - More flip-flops
  - It normally has simple combinational logic for next state transitions and output signals. It is suitable for high performance system design.
  - It is unlikely to have glitches.
- FPGAs have plenty of flip-flops. Thus, it is preferred to use one-hot encoding in FPGA FSM implementations.

Possible Glitches in Binary-Encoded FSMs

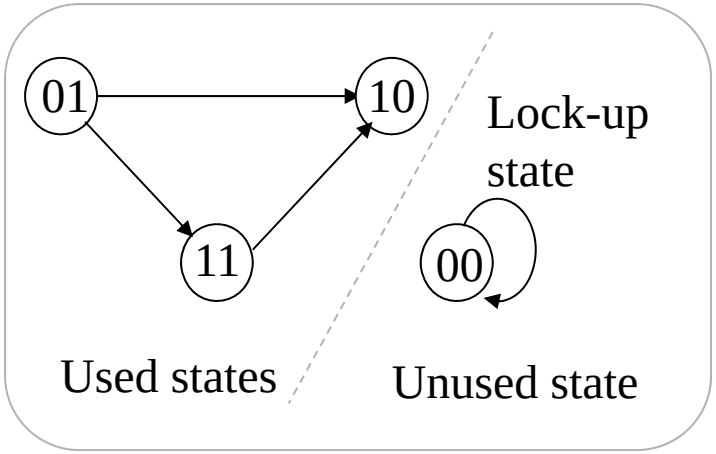


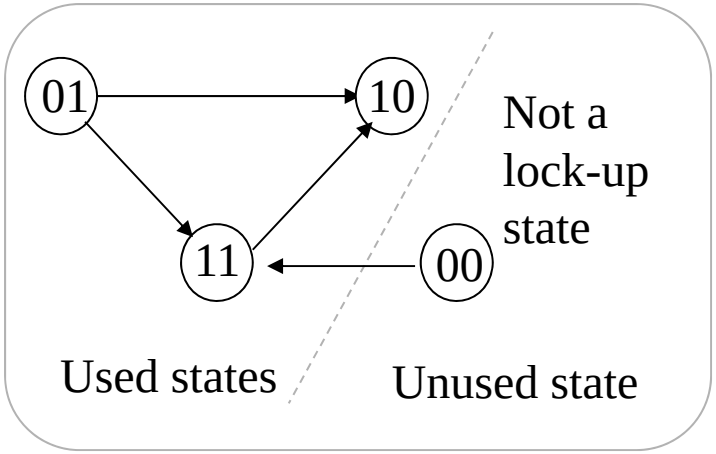
- If "11" is a legal state and certain operations are associated with state "11", the glitch may cause unwanted circuit operation
- In one-hot encoding, "11" is not a legal state and, hence, it will not trigger unwanted circuit operations.



Possible Lock-up States in Binary-Encoded FSMs

- When unused states exist in a binary-encoded FSM, make sure there are no lock-up states.
  - — Example: for a three state FSM, it needs two D flip-flops to implement binary encoding scheme. Assume 01, 10, 11 are the three used states. 00 is the unused state, make sure that FSM will not be trapped in 00 state.

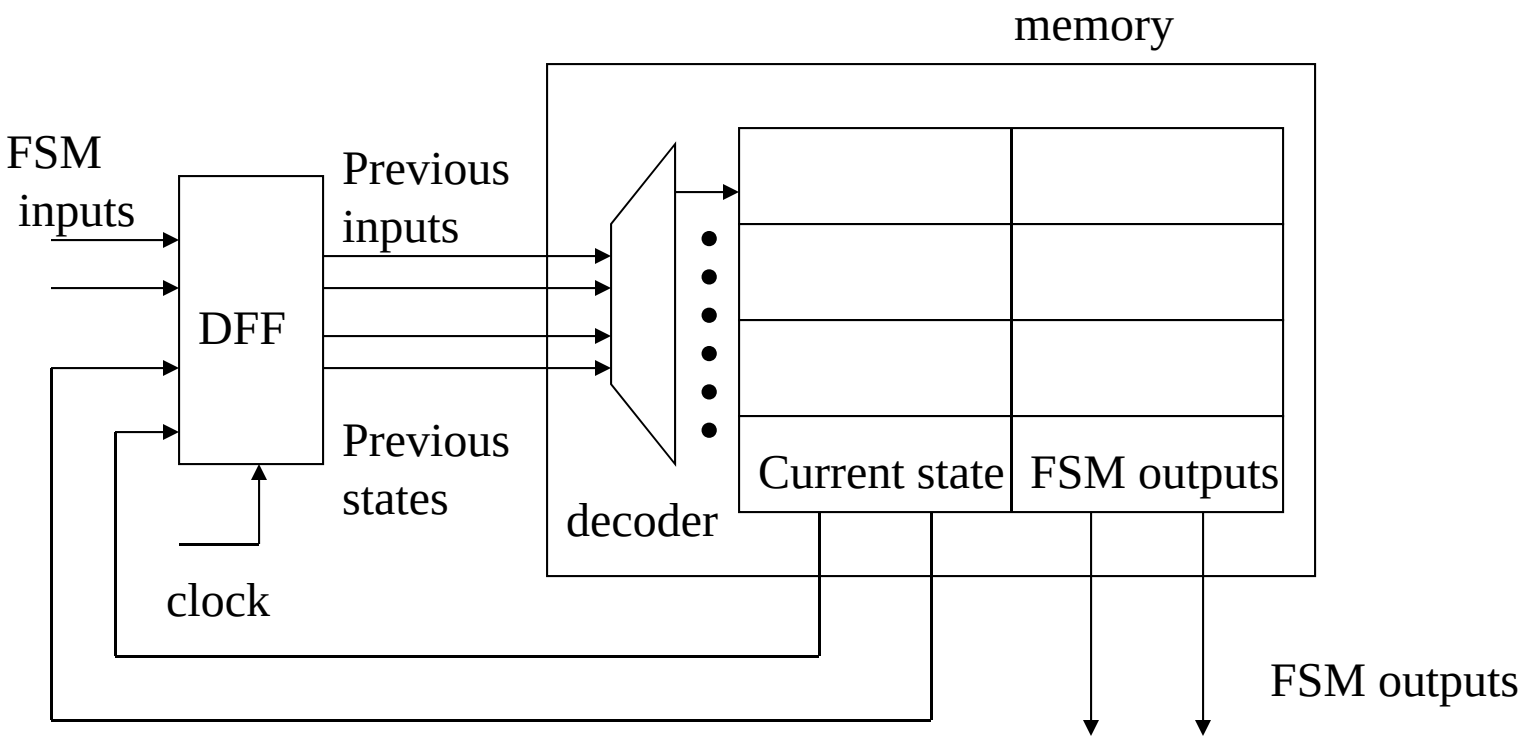




- To avoid lock-up states, make sure the FSM will eventually move from any unused state to an used state. Another method is to use reset (or set) signals to reset (or set) FSM to an used state (initial state) after power-up.

Implement Complex FSM using embedded Memories

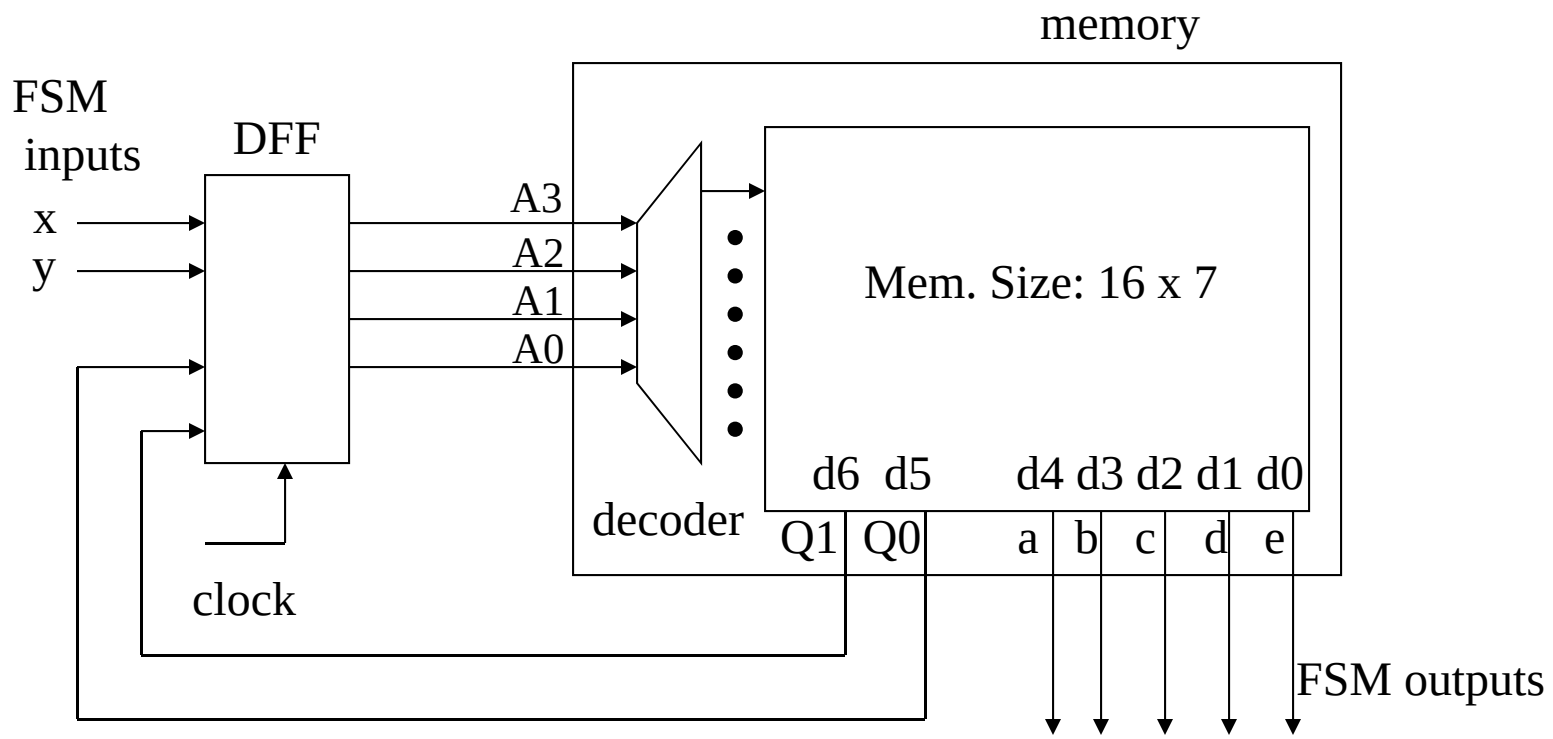
- Latest FPGAs often contain embedded memories. Complicated FSMs can be implemented by using the embedded memories.



It is similar to the microprogram mechanism used in CISC computers.

Example: memory-based FSM implementation

- Example FSM on slide 8-5
- Use the state encoding shown on slide 8-7
- Circuit hardware (for clocked memory, the DFFs are not needed)



Example: memory-based FSM implementation

- Date stored in memory (note the difference from the table on slide 8-7)

Address				Memory Content						
A3(x)	**A2(y)**	**A1(Q1)**	**A0(Q0)**	d6(Q1)	d5(Q0)	d4(a)	d3(b)	d2(c)	d1(d)	d0(e)
0	x	0	0	1	1	0	0	0	0	0
0	x	0	1	1	0	1	0	0	1	0
0	x	1	0	1	1	0	0	0	0	0
0	x	1	1	1	1	0	0	0	0	0
1	0	0	0	0	1	0	1	0	1	1
1	0	0	1	1	0	1	0	0	1	0
1	0	1	0	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	1	0	1	0	0	1	0
1	1	0	1	1	0	1	0	0	1	0
1	1	1	0	1	0	1	0	0	1	0
1	1	1	1	1	1	0	0	0	0	0